Simple PCPs with Poly-log Rate and Query Complexity*

Eli Ben-Sasson
Computer Science Department
Technion - Israel Institute of Technology
Haifa, Israel, and
Toyota Technological Institute
Chicago, IL 60637
eli@eecs.harvard.edu

Madhu Sudan †
Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139
madhu@mit.edu

ABSTRACT

We give constructions of probabilistically checkable proofs (PCPs) of length $n \cdot \text{poly}(\log n)$ (to prove satisfiability of circuits of size n) that can verified by querying $\text{poly}(\log n)$ bits of the proof. We also give constructions of locally testable codes (LTCs) with similar parameters.

Previous constructions of short PCPs (from [5] to [9]) relied extensively on properties of *low* degree *multi*-variate polynomials. In contrast, our constructions rely on new problems and techniques revolving around the properties of codes based on *high* degree polynomials in *one* variable (also known as Reed-Solomon codes). We show how to convert the problem of verifying the satisfaction of a circuit by a given assignment to the task of verifying that a given function is close to being a Reed-Solomon codeword, i.e., a univariate polynomial of specified degree. This reduction is simpler than the corresponding steps in previous reductions, and gives a new alternative to using the popular "sum-check protocol". We then give a new PCP for the special task of proving that a function is close to being a Reed-Solomon codeword. This step of the construction is by a self-contained recursion, and the only ingredient needed in the analysis is the bi-variate low-degree test of Polischuk and Spielman [27].

Note that our constructions yield LTCs first, which are then converted to PCPs. In contrast, most recent constructions go in the opposite (and less natural) direction of getting LTCs from PCPs.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of algorithms and problem complexity

*This work was done while both authors were at the Radcliffe Institute for Advanced Study, Cambridge, MA.

†Supported in part by NSF Award CCR-0312575.

STOC'05, May 22-24, 2005, Baltimore, Maryland, USA.

General Terms

Theory

Keywords

PCP, PCPP, Probabilistically Checkable Proofs, Locally testable codes

1. INTRODUCTION

Probabilistically Checkable Proof (PCP) systems [15, 3, 2] (a.k.a. Holographic Proofs [5]) are proof systems that allow efficient probabilistic verification based on probing few bits of a proof. Formally, a PCP system is given by a verifier, called a PCP verifier, that probabilistically queries a few bits of a purported proof of a claimed theorem and accepts valid proofs of true theorems with probability one, while accepting any claimed proof of false assertions with low probability, say at most 1/2. The celebrated PCP Theorem [3, 2] asserts that there exists a PCP verifier probing the proof in just a constant number of bits, and it turns out that probing three bits suffices (cf. [21], see also [19]). Furthermore the proof needed by this verifier is only polynomially larger than any classical proof. Such query efficient proofs translate to strong non-approximability results for many combinatorial optimization problems (cf. [8, 7, 21, 19, 29]).

Somewhat surprisingly, PCPs are rarely appreciated for their positive effects: i.e., as methods of transforming proofs into extremely efficiently verifiable formats. (Instead it is the negative implications to combinatorial optimization that dominate their study.) In principle, PCPs could form the semantic analog of error-correcting codes: Error-correcting codes are used to preserve data for long periods of time; PCPs may be used to preserve data, with a promise of integrity with respect to any fixed Boolean property, for long periods of time. However such uses seemed to be ruled out by current PCP constructions which are *too long* and *too complex*. This forms the motivation of our work, which tries to find *shorter and simpler PCPs*.

Optimizing the length of the new NP witness has already been the focus of [5, 27, 20, 18, 11, 9]. In addition to the inherent motivation mentioned above, the length of PCPs also plays an important role in their use in cryptography (e.g., CS-proofs [22, 26] and their applications [6, 12]) and is closely related to constructions of locally testable codes [18, 11, 9]. Simplifying PCP constructions has long been a goal within the study of PCPs, though little progress has been achieved in this direction so far. Only recently, Dinur and

Reingold [14], took some first steps towards deriving a PCP construction purely combinatorially. (Our efforts run in the opposite direction, by invoking more algebra to construct simple PCPs.)

PCPs. Our main result is a PCP construction that blows up the NP-witness length by only a poly-logarithmic factor and can be verified by querying a poly-logarithmic number of bits of the proof. PCPs are traditionally measured by their randomness and query complexity with PCP$[r, q]$ being the class of all languages L that have PCP verifiers tossing $r(n)$ coins, running in polynomial time in n and querying $q(n)$ bits to verify proofs of membership of theorems of the form $x \in L$, where $n = |x|$. In this notation, our main theorem is stated formally below. (Throughout this paper, all logarithms are to the base two).

THEOREM 1 (EFFICIENT PCPS). SAT *has a PCP verifier that on inputs of length n tosses $\log(n \operatorname{poly}(\log n))$ coins, makes $\operatorname{poly}(\log n)$ queries to a proof oracle of length $n \operatorname{poly}(\log n)$, runs in time $n \operatorname{poly}(\log n)$ and has perfect completeness and soundness at most $\frac{1}{2}$. Formally,*

$$\text{SAT} \in PCP_{1,\frac{1}{2}}[\log(n \operatorname{poly} \log n), \operatorname{poly} \log n]$$

In contrast, the recent results of Ben-Sasson et al. [9] give proofs of length $n \cdot \exp(\operatorname{poly} \log \log n)$ with a query complexity of $\operatorname{poly} \log \log n$. Thus, while the query complexity of our PCPs is higher than that of most recent results, the proof size is significantly smaller.

PCPs of Proximity. The results of [9] refer to a stronger notion of PCPs, called *PCPs of Proximity* (denoted PCPPs). Whereas a PCP for a Boolean circuit only guarantees the existence of a satisfying assignment, a PCPP guarantees that an assignment (provided as a separate oracle from the proof) is close to some assignment that satisfies the circuit. Theorem 1 implies efficient PCPPs for any circuit as in [9], but with shorter proofs. The formal statement and proof of the following Theorem are omitted due to space constraints.

THEOREM 2 (PCPPS FOR SAT (INFORMAL)). *For every constant $\delta \in (0, 1)$ there exists a verifier V_{SAT} (i.e. a randomized polynomial time oracle Turing machine) that on input a circuit ϕ of size n, tosses $\log(n \operatorname{poly}(\log n))$ coins, makes $\operatorname{poly}(\log n)$ queries to an assignment oracle α of length n' and proof oracle π of length $n \operatorname{poly}(\log n)$. The verifier has perfect completeness, i.e. accepts with probability one satisfying assignments (when accompanied by a suitable proof). Regarding soundness, any assignment that is δ-far from satisfying ϕ is rejected with probability $\geq 1/2$, no matter what proof is provided for it.*

Locally Testable Codes:. PCPs typically go hand-in-hand with a form of error-correcting codes, called Locally Testable Codes (cf. [18] and references therein), whose noise rate can be estimated in sub-linear time. Specifically these codes have an associated verifier that makes few accesses to an oracle describing an input word, and accepts codewords with probability one, while rejecting words that are δ-far from the code with constant probability (say 1/2). Once again, important parameters of these codes are their rate i.e., the ratio of the length of the message to the length of the codewords, the proximity parameter denoted δ above, and the query complexity of the verifier, i.e., the number of queries made by the verifier. In this work we also give LTCs with parameters comparable to those of our PCPs. The proof of the following Theorem follows from [9, Section 4.1], and will appear in the full version.

THEOREM 3 (EFFICIENT LTCS). *There exists a universal constant c' such that for every field $\mathbb{F}$ of characteristic two and every integer n that is a power of two, there exists an explicitly constructible code over alphabet $\mathbb{F}$, mapping messages of length $n/2$ to codewords of block-length $n \log^{c'} n$. The resulting code is locally testable with* query complexity *$\log^{c'} n$ and* proximity parameter *$\log^{-c'} n$. The* rate *of this code is $\frac{1}{2} \log^{-c'} n$.*

Note it is straightforward to get a binary code from the above by encoding elements of $\mathbb{F}$ by some binary code of block length $O(\log |\mathbb{F}|)$. The rate and relative distance of the resulting binary code are within constant factors of the $\mathbb{F}$-ary code above; and the query complexity of the local test increases by a $\log |\mathbb{F}|$ factor.

We highlight the fact that we first get LTCs as above and the PCPs are derived as a consequence. While the early work of Babai et al. [5] also had this feature, constructions of smaller LTCs (in particular those in [18, 11, 9]) reverse this direction, getting PCPs first, and then deriving LTCs as a consequence. Our work thus achieves one of the goals associated with LTCs, that direct constructions of LTCs may offer some benefits/insights to PCP constructions.

Our techniques. Our construction, while being algebraic, is significantly different from prior PCP constructions, so as to make the techniques interesting in their own right. Most previous constructions (with the exception of [14]) start with a PCP based on the properties of multivariate polynomials over some finite field. Some key ingredients in such constructions are (1) A *low-degree test*: A method to test if a function given by an oracle is *close* to being a low-degree multivariate polynomial. (2) A *self-corrector*: An efficient procedure to compute the value of a multivariate polynomial at a given point, given oracle access to a polynomial that is close to this function. (3) A *Zero-tester*: An efficient procedure to verify if a function (given by an oracle to compute it) is close to a multivariate polynomial that is zero on every point in a prespecified subset of its domain. (4) A reduction from verifying satisfiability to zero-testing. Typical solutions to the above problems yield a query complexity that grows polynomially in the number of variables and the degree of the multivariate polynomial. This query complexity, if too large, is then reduced by a new layer of techniques referred to as proof composition.

Our solution follows a similar outline (almost, as we do not need a self-corrector), however, we work (for the most part) only with univariate polynomials. This forms the essence of our technical advantage, giving us shorter PCPs. The length of PCPs is well-known to grow with the number of variables used in them, and reducing this number was an obvious way to try to reduce their lengths. However, the degree of the associated polynomials reduces as the number of variables increase and since solutions to steps (1)-(3) above had query complexity polynomial in the degree of the associated polynomial, previous solutions needed to use large number of

variables to achieve any non-trivial reduction in the number of queries. In our case, we propose analogous questions for *univariate* polynomials and give extremely efficient solutions (in terms of the degree of the associated polynomial). We describe our solutions to the steps in reverse order.

Reducing* SAT *to Univariate Zero Testing. We start with the reduction from satisfiability to testing zeroes of polynomials (step (4) above). The typical outcome of this part is a transformation from a SAT formula ϕ to a constraint C (on pairs of polynomials) and subsets H_1 and H_2 of the multivariate domains such that ϕ is satisfiable if and only if there exist polynomials P_1, P_2 that are zeroes on H_1, H_2 respectively and furthermore $C(P_1, P_2)$ holds. (To enable "easy verification", $C(P_1, P_2)$ needs to be of a special form, but we won't get into this now.) In general, these reductions are naturally simple, and so is ours. In Section 4 we describe how one can get some natural reductions from NP-complete problems to problems about testing zeroes of polynomials. The final result we use (Theorem 9) is somewhat more complex only due to our goal of extreme length efficiency.

Univariate Zero Testing. Next we move to the zero-testing problem (step (3) above). For this part we provide an extremely simple proof which reduces this question to two univariate "low-degree testing" questions, along with a natural consistency test between the two polynomials. Our query complexity is a *constant* independent of the degrees of the polynomials we are working with! Furthermore, it directly reduces zero-testing to "low-degree testing" while most previous solutions relied on some form or other of the "self-correcting" question. Put together our two steps above give an efficient reduction from verifying NP-hard statements to testing the degree of a univariate function. *Furthermore, these reductions add only a constant number of queries to the query complexity of the low-degree testing protocol!* This highlights the importance of the low-degree testing problem for univariate polynomials, which we describe below. Our techniques allow us to revisit the multi-variate zero testing problem and offer a new alternative to the commonly used "sum-check protocol" of [25] (Section 6).

Reed-Solomon codes and Proofs of Proximity:. The central problem at the heart of our PCPs is the following: Given a finite field $\mathbb{F}$, a degree bound d and oracle access to a function $f : \mathbb{F} \to \mathbb{F}$, test if f is (close to) a polynomial of degree at most d. Specifically, if f is a degree d polynomial then our test must accept. On the other hand, if f is δ-far from every degree d polynomial (i.e., the value of f needs to be changed on at least δ fraction of the points in $\mathbb{F}$ to get a degree d polynomial), then the test must reject with high probability. And it should do all this while querying the oracle for f as few times as possible. The class of functions derived by evaluating polynomials of a specified degree over a field is well-known as the *Reed-Solomon code.* Our goal is thus to provide an efficient test for membership in this code. It is easy to see that, as such, the problem above allows no efficient solutions: A tester that accepts all polynomials with probability one, must probe the value of f on at least $d + 2$ places before it can reject any function, and this is too many queries for our purpose. However, one can attempt to use some auxiliary information π about f that might allow for more efficient tests. Such auxiliary information is what is referred to as "Probabilistically Checkable Proof of Proximity" (PCPP) in the work of Ben-Sasson et al. [9] and as "Assignment Testers" in the work of Dinur and Reingold [14]. Both works define such a concept generically, for testing proximity to *possessing* any property, while we need it only for the *special case* of testing proximity to Reed-Solomon codewords. In terms of PCPPs, our new task is thus to design a tester that makes few oracle queries to a pair of oracles (f, π) (say both return elements of $\mathbb{F}$ as answers), with the following properties: If f is a degree d polynomial, there must exist a valid proof π so that (f, π) is always accepted by the tester. If f is δ-far from every degree d polynomial then for every π, (f, π) must be rejected with high probability.

Moderately efficient solutions to this problem can be obtained as a direct consequence of the final theorem of Ben-Sasson et al. [9], who give length efficient proofs for any property (relative to the time it takes to verify the property deterministically). However, such a solution would neither be simple, nor as short as we desire. (but it does clarify that our goal of making $o(d)$ queries is attainable!)

Our main technical result (Theorem 4) is a proof of proximity for the Reed-Solomon code (denoted the RS-code). This proof has length $O(n \operatorname{poly} \log n)$ for RS codes over a field $\mathbb{F}$ of cardinality n and characteristic two. We also describe some variations, such as PCPP for RS codes over certain prime fields (Theorem 5), but these are not needed for our final PCP results. The construction of these PCPPs is self-contained and relies only on basic algebra. Our proof of proximity consists of an encoding of an efficient FFT-like evaluation of the low degree polynomial. The only complex ingredient in our analysis is the (black-box) use of the analysis of Polishchuk and Spielman [27] of a natural low-degree test for bivariate polynomials.

We stress that the construction of our PCPP for the RS code is similar the PCPP proof composition of [14, 9]. The main difference is that previous works used composition of PCPPs for *general* circuits, whereas we use *special purpose* PCPPs (only) for Reed-Solomon codes. As discussed earlier, obtaining PCPPs for this restricted case suffices for obtaining PCPs for SAT (and proving Theorem 1).

Organization of this paper:. In Section 2 we define RS codes and PCPs of Proximity formally, and give an overview of our PCP of Proximity for RS codes over fields of characteristic two. In Section 3 we present our efficient zero-tester, whose query complexity is independent of the degrees of polynomials being tested (modulo the complexity of PCPPs for RS codes). In Section 4 we give the reduction from SAT to algebraic problem. The resulting PCP is described in Section 5, proving Theorem 1. In Section 6 we generalize the univariate zero testing protocol to the multivariate case.

2. PROOFS OF PROXIMITY FOR REED-SOLOMON CODES

We start with some basic notation used in the rest of the paper. Unless stated otherwise, we measure distance between $x, y \in \Sigma^n$ using the normalized Hamming distance, i.e. $\Delta(x, y) \triangleq \Pr_{i \in [n]}[x_i \neq y_i]$. Let C be a subset of Σ^n. (Often, but not always, in this paper Σ will be a field and C a linear error correcting code.) The distance of x from C (denoted $\Delta(x, C)$) is the minimal distance between x and a

member of C. If C is empty we define $\Delta(x, C) = 1$ for every x. We say x is δ-far from C if $\Delta(x, C) > \delta$, and otherwise x is δ-close to C.

Proofs of Proximity:. Here we consider the task of giving efficiently verifiable proofs of the statement "$x \in C$", where the verifier makes few queries into x *and the proof.* Such a verification task is necessarily probabilistic, and can only guarantee (upon acceptance) that x is δ-close to C. This notion was introduced in [9] as "Probabilistically Checkable Proofs of Proximity" (PCPP), and independently by [14] as "Assignment Tester". Our definition below is a slight variant on [9, Definition 2.3].[1]

DEFINITION 1 (PCPP). *A set $C \subseteq \Sigma^n$ is said to have a* Probabilistically Checkable Proof of Proximity *(PCPP) over alphabet Σ of* length $\ell(n)$*, with* query complexity $q(n)$*,* randomness $r(n)$*, perfect completeness and* soundness $s(\cdot, n)$*, if there exists a polynomial time randomized* verifier V *with oracle access to a pair $(x, \pi) \in \Sigma^{n+\ell(n)}$ such that V tosses $r(n)$ coins, makes $q(n)$ queries into (α, π), and outputs* accept *or* reject *with the following guarantees:*

Completeness: *If $x \in C$ then $\exists \pi \in \Sigma^{\ell(n)}$ such that verifier accepts (x, π) with probability 1.*

Soundness: *If $\Delta(x, C) \geq \delta$ then $\forall \pi \in \Sigma^{\ell(n)}$, verifier rejects (x, π) with probability $\geq s(\delta, n)$.*

As such PCPPs can be defined for any property $C \subseteq \Sigma^n$, but we care about it only for the case of the Reed-Solomon codes, which we define next.

DEFINITION 2. *For $P(z)$ a polynomial over a field $\mathbb{F}$ and $S \subseteq \mathbb{F}$ define its* evaluation table *over S to be $\langle P(z) \rangle_{z \leftarrow S} \triangleq \langle P(s) : s \in S \rangle$.* [2] *The* Reed-Solomon code *of degree d over $\mathbb{F}$, evaluated at S is defined as*

$$\mathrm{RS}(\mathbb{F}, S, d) = \{\langle P(z) \rangle_{z \leftarrow S} : P(z) = \sum_{i=0}^{d-1} a_i z^i, a_i \in \mathbb{F}\}.$$

The fractional degree *of such a code is $d/|S|$.*

The main result of this section gives a PCPP for RS-Codes over some nice fields and nice sets S. First we state the result for RS-codes over fields of characteristic two. (Other cases including certain prime fields are described below). The proof generalizes to arbitrary constant characteristic but we state and prove it for characteristic two, for the sake of simplicity and since this suffices for our PCP applications. Recall that a field $\mathbb{F}$ of cardinality 2^ℓ can be viewed as a vector space over $GF(2)$ of dimension ℓ. We say that $S \subseteq \mathbb{F}$ is "linear" if it is a linear subspace of this ℓ-dimensional space. (Equivalently, S is linear if for every $\alpha, \beta \in S$ we have $\alpha + \beta \in S$.)

THEOREM 4 (BINARY RS PCPP). *There exist universal constant $c \geq 1$ such that for every field $\mathbb{F}$ of characteristic two, every linear $S \subseteq \mathbb{F}$ with $|S| = n$, and every $d \leq n$, the Reed-Solomon code $\mathrm{RS}(\mathbb{F}, S, d)$ has a PCPP over alphabet $\mathbb{F}$ with the following parameters:*

- Proof length $\ell(n) \leq n \log^c n$.
- Randomness $r(n) \leq \log n + c \log\log n$.
- Query complexity $q(n) = O(1)$.
- Soundness $s(\delta, n) \geq \delta / \log^c n$.

Furthermore, there exists a Turing machine that on input $(\mathbb{F}, S, d)$ as above, outputs the (description of the) verifier for $\mathrm{RS}(\mathbb{F}, S, d)$ in time that is polynomial in $|\mathbb{F}|$.

Before sketching the proof of Theorem 4, we state its analog for fields that have a "nice" multiplicative sub-group. We point out there are infinitely many prime fields of this form, and given n, a suitable prime (of size polynomial in n) can be found efficiently. This claim is a corollary of a fundamental theorem in Number Theory by Linnik [24] (For more details, see `http://mathworld.wolfram.com/LinniksTheorem.html`). In turn, Theorem 5 could be used to obtain a different construction of our efficient PCPs (Theorem 1).

Let $\mathbb{F}^\star$ denote the cyclic multiplicative group of $\mathbb{F}$. Let the order of an element $\omega \in \mathbb{F}^\star$ be the smallest positive integer n such that $\omega^n = 1$. We refer to an integer n as a *power of two* if $n = 2^k$ for integer k. The multiplicative group generated by ω is $\langle \omega \rangle \triangleq \{\omega^0, \omega^1, \ldots, \omega^{n-1}\}$.

THEOREM 5 (MULTIPLICATIVE RS PCPP). *There exists a universal constant $c \geq 1$ such that the following holds. Let $\omega \in \mathbb{F}^\star$ be an element of order n in the finite field $\mathbb{F}$, where n is a power of two, let $S = \langle w \rangle$ and let $d < n$ be an integer. The Reed-Solomon code $\mathrm{RS}(\mathbb{F}, S, d)$ has a PCPP over alphabet $\mathbb{F}$ with the following parameters:*

- Proof length $\ell(n) \leq n \log^c n$.
- Randomness $r(n) \leq \log n + c \log\log n$.
- Query complexity $q(n) = O(1)$.
- Soundness $s(\delta, n) \geq \delta / \log^c n$.

Furthermore, there exists a Turing machine that on input $(\mathbb{F}, S, d)$ as above, outputs the (description of the) verifier for $\mathrm{RS}(\mathbb{F}, S, d)$ in time that is polynomial in $|S| + \log |\mathbb{F}|$.

Examination of the proof of Theorem 5 shows it can be derived for any $\langle \omega \rangle$ of size n that is $\mathrm{poly}(\log n)$-smooth, i.e. all prime factors of n are at most $\mathrm{poly}(\log n)$. Alternatively, the proof can be modified to obtain similar PCPPs for RS-codes over fields of characteristic $\leq \mathrm{poly}(\log n)$ (in this setting, we use the additive structure of the field as in the proof of Theorem 4). For simplicity, we state the Theorem only for the multiplicative case of a 2-smooth n.

The soundness in Theorems 4,5 can be boosted to an arbitrary constant less than 1 (for any fixed $\delta > 0$), using any averaging (a.k.a. oblivious) sampler (cf. [17]). In the statement of the following Corollary 6, we say $\mathrm{RS}(\mathbb{F}, S, d)$ has a PCPP with soundness half for proximity parameter δ, if every word that is δ far from $\mathrm{RS}(\mathbb{F}, S, d)$ is rejected with probability at least half (the choice of half is arbitrary and could be replaced by any constant smaller than one).

[1] We require the soundness be a function of the proximity, whereas [9] only needed the soundness to be large whenever the distance is large.

[2] Strictly speaking S should be a sequence for this definition to work. We'll assume some canonical ordering of elements of S and assume $z \leftarrow S$ enumerates z according to this ordering.

COROLLARY 6. *There exists a universal constant $c \geq 1$ such that for any $\delta \in (0,1)$ the following holds. For $\mathbb{F}, S, d$ be as in the statement of Theorem 4 or Theorem 5, the Reed-Solomon code* $\mathrm{RS}(\mathbb{F}, S, d)$ *has a PCPP over alphabet* $\mathbb{F}$ *with* proof length $\ell(n) \leq n \log^c n$, randomness $r(n) \leq \log n + c \log\log n$, query complexity $q(n) = \log^c n/\delta$, *and* soundness *half for* proximity parameter δ.
Furthermore, there exists a Turing machine that on input $(\mathbb{F}, S, d)$ as above and $\delta \in (0,1)$ represented in binary notation by k bits, outputs the (description of the) above mentioned verifier in time that is polynomial in $|S| + \log|\mathbb{F}| + k$.

A full proof of Theorems 4,5 will appear in the full version of this paper. A proof of Theorem 5 (and the PCPs resulting from it) appears in our preliminary report [10]. Here we only sketch the proof of Theorem 4, due to space limitations.

PROOF. (Sketch, of Theorem 4): The PCPP is essentially built from first principles: At a high level, we attempt an elementary reduction from the task of testing a univariate polynomial to the task of testing a bivariate polynomial of significantly smaller degree. We then invoke an analysis of a "bivariate low-degree test" by Polishchuk and Spielman [27], which reduces the task of testing bivariate polynomials back to the task of testing univariate polynomials, of much smaller degree than the original. Recursing on this idea leads to the full test. We note that crucial to our obtaining short PCPPs, is the evaluation of the bi-variate polynomial on a carefully selected, algebraically structured, subset of points. This set is very different from the sets typically used in previous PCP constructions (e.g. [5, 2, 13]), which are product sets (usually consisting of the whole field).
We start by considering the polynomial $P(z)$ of degree $< n/8$ evaluated on the linear space $L \subset GF(2^\ell)$ of cardinality n, and address the task of "testing" it. Our main idea is that for any polynomial $q(z)$ of degree $\approx \sqrt{n}$, we can define a bivariate polynomial $Q(x,y)$ of degree $\approx \sqrt{n}$ in each variable, that "captures" all the information of P. Specifically, we can reconstruct P from Q using the identity $P(z) = Q(z, q(z))$. The presentation of P of degree $\approx n$ as a bivariate polynomial Q of individual degree $\approx \sqrt{n}$ is useful, because testing of bivariate polynomials reduces to testing of univariate polynomials of roughly the same degree using well-known "low-degree tests" and their analysis (cf. [27]). This leads us to the hope that Q might provide a good "proof" that P is of low-degree. More to the point, to prove that a table of evaluations of P corresponds to the evaluations of a polynomial of low-degree, the prover can provide a table of evaluations of a bivariate polynomial Q, prove that Q has degree $\sqrt{n}$ in each variable, and then prove that Q is consistent with the table of evaluations of P.
To completely describe the above approach, all we need to do is describe which set of points we will specify Q on, so as to achieve both tasks: *(i)* verifying that Q has low-degree, and *(ii)* that it is consistent with P. However this leads to conflicting goals. In order to test that Q has low-degree, using a bivariate verifier, we need to know its values on some subset $X \times Y$ where $X, Y \subseteq GF(2^\ell)$. To make this efficient, we need to make $|X|, |Y| \approx \sqrt{n}$. On the other hand to test its consistency with P, the natural approach is to ask for its values on the set $T = \{(z, q(z)) | z \in L\}$. Unfortunately the set T, which depends on the selection of $q(z)$, is far from being of the form $X \times Y$. (For starters, the projection of T onto its first coordinate has cardinality n while we would like this projection to be of cardinality $O(\sqrt{n})$.)
This discrepancy (between T and cross-product sets) seems to kill this approach entirely, however it turns out it can be salvaged. To do so, we choose $q(z)$ to be a special *linearized* polynomial (cf. [23, Chapter 3, Section 4]) that we now describe. A polynomial q over $GF(2^\ell)$ is said to be linearized if the mapping defined by it is $GF(2)$-linear (i.e. $q(x+y) = q(x) + q(y)$ for every $x, y \in GF(2^\ell)$). Let $\tilde{L}_0, \tilde{L}_1$ be arbitrary linear subspaces of L such that L is the direct sum of $\tilde{L}_0, \tilde{L}_1$ and $\dim(\tilde{L}_0) = \lfloor \dim(L)/2 \rfloor, \dim(\tilde{L}_1) = \lceil \dim(L)/2 \rceil$ (so $|\tilde{L}_0|, |\tilde{L}_1| \approx \sqrt{|L|}$). We take $q(z)$ to be the unique monic polynomial of minimal degree whose roots are precisely the elements of $\tilde{L}_0$, i.e.

$$q(z) \triangleq \prod_{\tilde{\alpha} \in \tilde{L}_0} (z - \tilde{\alpha})$$

The polynomial $q(z)$ is linearized and as such has nice properties that will be crucial in our analysis:

- q defines a linear map over L.
- The kernel of (the linear map defined by) q is $\tilde{L}_0$.
- The image of L under (the linear map defined by) q is a linear space denoted L_1, and $\dim(L_1) = \dim(\tilde{L}_1)$.
- The polynomial q can be computed and evaluated on elements of the field in polynomial time in $|\mathbb{F}|$.

With this information in hand we notice $q(z)$ partitions T into $\approx \sqrt{n}$ product sets as follows. For $\tilde{\beta} \in \tilde{L}_1$ let $\beta = q(\tilde{\beta})$, and notice $\beta \in L_1$. Let $\tilde{A}_{\tilde{\beta}}$ be the affine shift of $\tilde{L}_0$ by $\tilde{\beta}$ (i.e. $\tilde{A}_{\tilde{\beta}} = \{\tilde{\alpha} + \tilde{\beta} : \tilde{\alpha} \in \tilde{L}_0\}$). Finally, let $T_{\tilde{\beta}} = \tilde{A}_{\tilde{\beta}} \times \{\beta\}$. Then, $q(z)$ partitions T into the disjoint union of the product sets $T_{\tilde{\beta}}$ for $\tilde{\beta} \in \tilde{L}_1$. (See Figure 1).
This suggests requesting the evaluation of Q on the set of points $(\tilde{L}_0 \times L_1) \cup T$, the cardinality of which is $\leq 2n$. (See

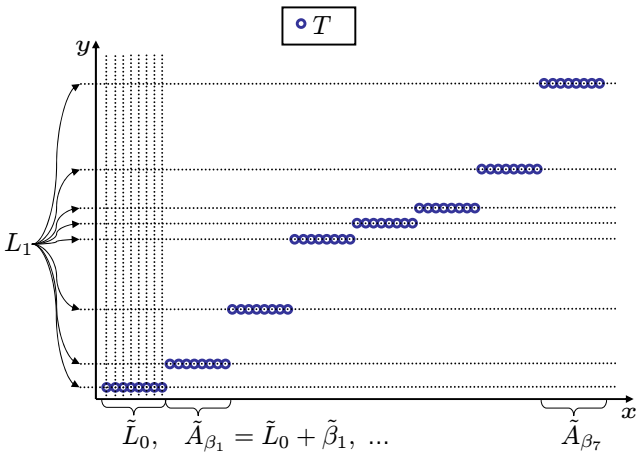

Figure 1: Here $\mathbb{F} = GF(2^6)$ is the field with 64 elements and q is a linearized polynomial of degree 8. We plot the set of points $T \subset \mathbb{F} \times \mathbb{F}$ defined by $T = \{(z, q(z)) : z \in \mathbb{F}\}$. Notice T can be partitioned into eight product sets, each set being a product of an affine shift of $\tilde{L}_0$ and some $\beta \in L_1$.

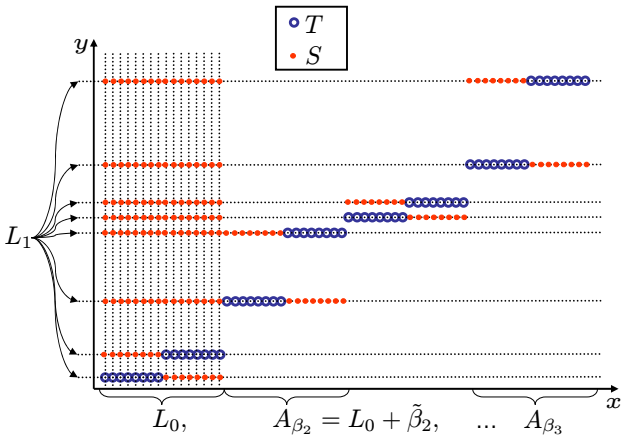

Figure 2: The proof of proximity for P is the evaluation of Q on the set of points S. Notice S has a large product set $L_0 \times L_1$, allowing for bi-variate low degree testing. Additionally, $S \cup T$ can be partitioned into eight row, each row being a linear space.

figure 2). With such an evaluation in hand we can use the subset $\tilde{L}_0 \times L_1$ to perform a bivariate low degree test, by testing proximity to the RS-code (of degree $\approx \sqrt{n}$) of a random row/column of this product set. Then we can test consistency of Q on T by reading $Q(x, \beta)$ for all points $x \in \tilde{L}_0 \cup \tilde{A}_{\tilde{\beta}}$. A crucial observation is that all of our tests verify proximity to Reed-Solomon codewords evaluated on *linear subspaces* of $GF(2^\ell)$. To see this notice $\tilde{L}_0$ and L_1 are linear spaces (by definition), and so is the set $\tilde{L}_0 \cup \tilde{A}_{\tilde{\beta}}$ (it is the space spanned by $\tilde{L}_0$ and $\tilde{\beta}$). We have reduced our original problem of size n to $O(\sqrt{n})$ identical problems of size $O(\sqrt{n})$. Our description so far leads to a proof of proximity of size $O(n)$ that can be tested by making $O(\sqrt{n})$ queries. However, the *robustness* of our tests can be used to decrease the query complexity further, at the price of increasing the proof length. Informally, robustness means the following. The distance of a function $f : (\tilde{L}_0 \times L_1) \cup T \to GF(2^\ell)$ from a low degree bivariate polynomial is bounded by (a constant times) the expected distance of an individual test (of size $\approx \sqrt{n}$) from the respective RS code. Thus, in order to test proximity to the RS code of size n it suffices to test proximity to RS codes of size $\approx \sqrt{n}$ (which can be done by testing proximity to the RS code of size $\approx n^{1/4}$, etc.). Applying this recursion a $\log \log n$ number of times reduces the degree to constant and gives us our proofs of length $n \operatorname{poly}(\log n)$.

The completes our sketch. The full proof (omitted due to space limitations) will appear in the full version. ∎

3. ZERO TESTING FOR UNIVARIATE POLYNOMIALS

Our previous section showed how to test proximity to the RS-code. However, in our PCP constructions we will need to verify that a function $f : S \to \mathbb{F}$ is close to a low degree polynomial p that *vanishes* on $H \subset \mathbb{F}$. This problem can also be cast as a PCPP problem for the vanishing RS-code:

$$\mathrm{RS}_H(\mathbb{F}, S, d) \triangleq \{\langle P(z)\rangle_{z \leftarrow S} : \deg(P) \leq d, \forall h \in H,\ P(h) = 0\}$$

Notice we do not require H to be a subset of S in the problem definition.

We remark that this problem is a special case of an m-variate zero testing problem often considered in the literature on PCPs (see Section 6). In the m-variate version, one is given evaluations of a polynomial on all points in $\mathbb{F}^m$ (thus $S = \mathbb{F}^m$) and would like to verify that it is zero on all of H^m. Typical solutions to this problem make $O(m \cdot |H|)$ queries, which would be too large for us.

For our case, if $|H| > d$ the problem becomes trivial, because the zero-polynomial is the only element of $\mathrm{RS}_H(\mathbb{F}, S, d)$. However, we are interested in the case where $|H| < d$. The natural solution (querying f on all/random $h \in H$) fails for two reasons. First, the size of H is larger than our desired query complexity, and furthermore all we gain from a proximity test for the RS code is that f is *close* to p, but it may still be the case that $f(\alpha) \neq p(\alpha)$ for one or more $\alpha \in H$.

The following Lemma reduces zero-testing to RS-proximity testing. We remark that a similar result holds for multivariate zero testing, and can be used to simplify previous PCP constructions (as described in Section 6).

LEMMA 7 (UNIVARIATE ZERO TESTING). *Suppose that $\mathrm{RS}(\mathbb{F}, S, d)$ has a PCP of proximity of length ℓ, query complexity q, randomness r, and soundness $s(\delta)$. Then for any $H \subset \mathbb{F}, |H| < |S|$, the vanishing RS-code $\mathrm{RS}_H(\mathbb{F}, S, d)$ has a PCP of proximity with length $\ell + |S|$, query complexity $q + 2$, soundness $\geq \min\{s(\delta/2), \delta/2\}$ and randomness $\max\{r, \log |S|\}$.*

Furthermore, the (description of the) verifier for $\mathrm{RS}_H(\mathbb{F}, S, d)$ can be computed in polynomial time in $|H|, \log |\mathbb{F}|$ and the (description of the) verifier for $\mathrm{RS}(\mathbb{F}, S, d)$.

As in the case of Theorems 4, 5 (and using the same techniques), we can boost the soundness using randomness-efficient samplers:

COROLLARY 8. *There exists a universal constant $c \geq 1$ such that for any $\delta \in (0, 1)$ the following holds. For $\mathbb{F}, S, d$ be as in the statement of Theorem 4 or Theorem 5 and $H \subset \mathbb{F}, |H| < |S|$, the vanishing Reed-Solomon code $\mathrm{RS}_H(\mathbb{F}, S, d)$ has a PCPP over alphabet $\mathbb{F}$ with* proof length *$\ell(n) \leq n \log^c n$,* randomness *$r(n) \leq \log n + c \log \log n$,* query complexity *$q(n) = \log^c n/\delta$, and* soundness *half for* proximity parameter *δ.*

Furthermore, there exists a Turing machine that on input $(\mathbb{F}, S, d, H)$ as above and $\delta \in (0, 1)$ represented in binary notation by k bits, outputs the (description of the) above mentioned verifier in time that is polynomial in $|S| + \log |\mathbb{F}| + k$.

PROOF. (of Lemma 7) Recall a polynomial $P(z)$ is zero on H iff the polynomial $g_H(z) \triangleq \prod_{h \in H}(z - h)$ divides it, i.e. $P(z) = g_H(z) \cdot \tilde{P}(z)$ for some polynomial $\tilde{P}, \deg(\tilde{P}) \leq d - |H|$. The verifier for $\mathrm{RS}_H(\mathbb{F}, S, d)$ has oracle access to the purported codeword $p : S \to \mathbb{F}$ and its proof, which is combined of two parts: *(i)* $\tilde{p} : S \to \mathbb{F}$ a supposed evaluation of $\tilde{P}$ on S; *(ii)* A proof $\tilde{\pi}$ of proximity of $\tilde{p}$ to $\mathrm{RS}(\mathbb{F}, S, d - |H|)$. Proof length is as claimed. The verifier operates as follows. First, it invokes the verifier for $\mathrm{RS}(\mathbb{F}, S, d - |H|)$ on input oracle $\tilde{p}$ and proof oracle $\tilde{\pi}$. Then, it picks a random $\alpha \in S$ (using same random coins used by the RS-verifier),

reads $p(\alpha)$ and $\tilde{p}(\alpha)$, and accepts iff $p(\alpha) = g_H(\alpha) \cdot \tilde{p}(\alpha)$. Notice $g_H(\alpha)$ can be computed in polynomial time by the verifier because H is given as an explicit input. Thus, the running time is as claimed, and so are the randomness and query complexity, by construction. Completeness follows by taking $\tilde{p}$ to be the evaluation of $\tilde{P}$, and taking $\tilde{\pi}$ to be its proof of proximity to $\mathrm{RS}(\mathbb{F}, S, d - |H|)$.

As to the soundness, if $\tilde{p}$ is $\delta/2$-far from $\mathrm{RS}(\mathbb{F}, S, d-|H|)$, the rejection probability of our verifier is at least $\delta/\operatorname{poly}\log n$, by the soundness of the verifier for $\mathrm{RS}(\mathbb{F}, S, d)$. Otherwise, the function $q : S \to \mathbb{F}$ defined by $q(\alpha) = \tilde{p}(\alpha) \cdot g_H(\alpha)$ is $\delta/2$-close to $\mathrm{RS}_H(\mathbb{F}, S, d)$. By assumption, p is $\delta/2$-far from q, hence the verifier rejects with probability at least $\delta/2$. (We assume n is sufficiently large, so $\operatorname{poly}\log n \geq 2$). This completes our proof. ∎

4. REDUCING SAT TO ALGEBRAIC PROBLEMS

In order to obtain length-efficient PCPs we reduce 3-SAT to an NP-complete *algebraic* constraint satisfaction problem (ACSP), that we now describe. First we consider a very natural reduction, that gives a polynomial length PCP. Then we sketch the steps used to obtain nearly linear blowup in our reduction.

A 3-CNF formula ϕ can be viewed as a characteristic function $\mathsf{Clause} : [n]^3 \times \{0,1\}^3 \to \{0,1\}$, denoting which clauses belong to ϕ. An assignment is a function $A : [n] \to \{0,1\}$, and it satisfies ϕ iff $\forall x, y, z \in [n], b_1, b_2, b_3 \in \{0,1\}$ we have

$$\mathsf{Clause}(x, y, z, b_1, b_2, b_3) = 0$$
$$\text{or } A(x) = b_1 \text{ or } A(y) = b_2 \text{ or } A(z) = b_3$$

Extending wlog the domain of Clause to $[n]^6$ (by associating $\{0,1\}$ with two elements of $[n]$ and fixing the function to zero on all new elements in the larger domain), the straight forward algebraic version of ϕ is a six-variate polynomial $\widehat{\mathsf{Clause}} : \mathbb{F}^6 \to \mathbb{F}$, of degree $\leq n$ in each variable and a set $H \subset \mathbb{F}, |H| = n \ll |\mathbb{F}|$ (We associate elements of H arbitrarily with those of $[n]$). An assignment is a pair of polynomials, a univariate $\widehat{A} : \mathbb{F} \to \mathbb{F}$ of degree $\leq n$ and a six variate $\widehat{B} : \mathbb{F}^6 \to \mathbb{F}$ of degree $\leq n$ in each variable. Such a pair satisfies $\widehat{\mathsf{Clause}}$ iff $\widehat{B}$ vanishes on H^6 and for all $x, y, z, b_1, b_2, b_3 \in \mathbb{F}$:

$$\widehat{B}(x, y, z, b_1, b_2, b_3) = \widehat{\mathsf{Clause}}(x, y, z, b_1, b_2, b_3) \cdot (\widehat{A}(x) - b_1) \cdot (\widehat{A}(y) - b_2) \cdot (\widehat{A}(z) - b_3)$$

The essential ingredients in this arithmetization are *(i)* Each of $\widehat{A}, \widehat{B}$ is low degree (compared to $|\mathbb{F}|$). *(ii)* The consistency of $\widehat{A}, \widehat{B}$ is verified by uniformly accessing each of them at random points. *(iii)* Testing $\widehat{A}$ is good reduces to (multivariate) zero testing of $\widehat{B}$. Indeed, this approach leads to simple PCPs of size $\approx n^6$.

Below we give a similar reduction, except that $\widehat{B}$ is univariate, which is essential for our nearly linear PCPs. (The price we pay for this length efficiency is that the reduction is a bit more complex to describe.) In contrast to a CNF, an assignment to our ACSP instance is a univariate polynomial (intuitively, it is the Reed Solomon encoding of the classical NP witness). The constraints of our problem are indexed by field elements (one constraint per element). Similar to a 3-CNF formula, each constraint depends on a finite number (nine) of assignment values (given by the assignment polynomial). However, the assignment entries we read are not arbitrary (as in a CNF), rather they are linear functions of the constraint index. Finally, the constraint polynomial itself is of small degree in the constraint index and multi-linear in the assignment values it reads.

DEFINITION 3 (UNIVARIATE ALGEBRAIC CSP). *The language L_{UACSP} has as its space of instances, tuples of the form $\phi_{\mathrm{ALG}} = (\mathbb{F}, \{\mathrm{AFF}_1, \ldots, \mathrm{AFF}_k\}, H, C)$, where $\mathbb{F}$ is a field, $\mathrm{AFF}_i = a_i x + b_i$ is an affine map over $\mathbb{F}$, $H \subset \mathbb{F}$ and $C : \mathbb{F}^{k+1} \to \mathbb{F}$ is a polynomial of degree at most $|H|$ in its first variable and is multi-linear of degree at most three in the remaining variables. An instance ϕ_{ALG} is in L_{UACSP} iff there exists a polynomial $A : \mathbb{F} \to \mathbb{F}$ of degree at most $|H|$ such that for every $x \in H$, $C(x, A(\mathrm{AFF}_1(x)), \ldots, A(\mathrm{AFF}_k(x))) = 0$.*

THEOREM 9 (SAT $\rightsquigarrow$ UNIVARIATE ALGEBRAIC CSP). *There exists a reduction from SAT to L_{UACSP} that given a Boolean formula ϕ of size n and integer $\ell > \log n + \log\log n + 14$, reduces ϕ in deterministic time $n \operatorname{poly}(\ell)$ to an instance $\phi_{\mathrm{ALG}} = (GF(2^\ell), \{\mathrm{AFF}_1, \ldots, \mathrm{AFF}_9\}, H, C)$, where $|H| \leq 20n \log n$.*

Very similar algebraic reductions are prevalent in many previous PCPs [5, 3, 2, 27, 30, 11, 9], starting with [5]. All previous reductions used multivariate polynomials in order to perform degree reduction. Namely, an assignment of length n is encoded by an m-variate polynomial of degree $\approx m \cdot n^{1/m}$ (allowing proximity testing with $n^{1/m}$ queries). Our reduction does not reduce the degree at all, in fact it slightly increases it. The PCPPs for the RS code allow us to tolerate this and verify proximity to high-degree polynomials with very small query complexity (logarithmic in the degree).

PROOF. (Sketch) We reduce our CNF to the univariate problem in two steps. Our first step is essentially identical to Polischuk and Spielman [27]. In this step we embed the CNF in a de Bruijn graph. Namely, we map variables and clauses of the input CNF to distinct vertices of the de Bruijn graph, and use the routing properties of this graph to connect clauses to their variables via vertex disjoint paths. Our intermediate problem asks for an assignment (to vertices of the graph) that satisfies the embedded CNF.

In the second step, we embed the de Bruijn graph in a sufficiently large field of characteristic two, and arithmetize the problem. Namely, we use a graph homomorphism injecting the de Bruijn graph into an "affine graph" (a generalization of a Cayley graph). This homomorphism allows us to form a new constraint satisfaction problem, where variables and constraints are labeled by elements of the field, and more importantly, the set of variables used in a constraint can be computed by a finite number of affine shifts of the constraint label. The resulting problem is our Univariate Algebraic CSP of Definition 3. (A formal proof will appear in the full version). ∎

5. PROOF OF MAIN THEOREM 1

Assume we are given a Boolean formula ϕ of size n. The verifier sets $\ell = \lceil \log n + \log\log n \rceil + 20$ and reduces ϕ to an instance $\phi_{\mathrm{ALG}} = (GF(2^\ell), \{\mathrm{AFF}_1, \ldots, \mathrm{AFF}_9\}, C, H)$ of L_{UACSP} using Theorem 9. Let $\mathbb{F} = GF(2^\ell)$ and notice $|H|/|\mathbb{F}| < 1/100$. Recall $|H| \leq 20n \log n$. Fix $\delta = 1/100$.

The proof oracle. Verifier has oracle access to two purported Reed-Solomon codewords and their PCPPs.

- An *assignment oracle* $p_A : \mathbb{F} \to \mathbb{F}$ and its proof of proximity to $\mathrm{RS}(\mathbb{F}, \mathbb{F}, |H|)$, denoted π_A, as defined in Corollary 6 (using proximity parameter δ).
- A *constraint oracle* $p_B : \mathbb{F} \to \mathbb{F}$ and its proof of proximity to $\mathrm{RS}_H(\mathbb{F}, \mathbb{F}, 4 \cdot |H|)$, denoted π_B, as defined in Corollary 8.

Let us calculate the size of the proof oracle. Notice $|p_A|, |p_B| = |\mathbb{F}| = O(n \log n)$ by construction, and $|\pi_A|, |\pi_B| = n \,\mathrm{poly}(\log n)$, by Corollaries 6, 8. The total proof length is $n \cdot \mathrm{poly}(\log n)$ even when measured in bits, because every element of $\mathbb{F}$ can be written using $O(\log n)$ bits.

Verifier's operation. Verifier tosses $\log n + O(\log \log n)$ coins to obtain a random string R, and accepts iff following three tests accept

- Invoke the verifier for the RS-code $\mathrm{RS}(\mathbb{F}, \mathbb{F}, |H|)$ described in Corollary 6 on input oracle p_A and proof oracle π_A, using proximity parameter δ and random string R.
- Invoke the verifier for the vanishing RS-code $\mathrm{RS}_H(\mathbb{F}, \mathbb{F}, 4 \cdot |H|)$ described in Corollary 8 on the input oracle p_B and proof oracle π_B, using proximity parameter δ and random string R.
- Select random $\beta \in \mathbb{F}$ (using random string R), and query $p_B(\beta)$ and $p_A(\textsc{Aff}_1(\beta), \ldots, p_A(\textsc{Aff}_9(\beta)))$. Accept iff

$$p_B(\beta) = C(\beta, p_A(\textsc{Aff}_1(\beta)), \ldots, p_A(\textsc{Aff}_9(\beta))).$$

The randomness, query complexity and running time of our verifier is as claimed, by Corollaries 6, 8 and by construction.

Completeness. By Theorem 9, if ϕ is satisfiable, there exists an assignment polynomial $P_A, \deg(P_A) \le |H|$ such that $P_B(x) \triangleq C(x, P_A(\textsc{Aff}_1(x)), \ldots, P_A(\textsc{Aff}_9(x)))$ vanishes on H. Recall $C : \mathbb{F}^{10} \to \mathbb{F}$ has degree $\le |H|$ in its first variable and degree three in the remaining ones. Hence, $\deg(P_B) \le |H| + 3 \cdot \deg(P_A) \le 4 \cdot |H|$. Let p_A, p_B be the evaluation of P_A, P_B on $\mathbb{F}$, respectively. Let π_A, π_B be the proofs of proximity to $\mathrm{RS}(\mathbb{F}, \mathbb{F}, |H|), \mathrm{RS}_H(\mathbb{F}, \mathbb{F}, 4 \cdot |H|)$ promised by (the completeness part of) Corollaries 6, 8, respectively. By these Corollaries and by construction, our verifier accepts with probability one, as claimed.

Soundness. Suppose ϕ is unsatisfiable. There are three cases to consider. *(i)* If p_A is δ-far from $\mathrm{RS}(\mathbb{F}, \mathbb{F}, |H|)$, then Corollary 6 implies the verifier rejects with probability $1/2$, and we are done. *(ii)* Similarly, if p_B is δ-far from $\mathrm{RS}_H(\mathbb{F}, \mathbb{F}, 4 \cdot |H|)$, then Corollary 8 implies the verifier rejects with probability $1/2$, and we are done.
(iii) Otherwise, let P_A be the unique degree $|H|$ polynomial that is δ-close to p_A and let P_B be the unique polynomial of degree $4|H|$ that is δ-close to p_B and vanishes on H. Let

$$P_C(x) = C(x, P_A(\textsc{Aff}(x)), \ldots, P_A(\textsc{Aff}_9(x))).$$

Since ϕ is unsatisfiable, Theorem 9 implies $P_B(x) \neq P_C(x)$. Since $\deg(P_B), \deg(P_C) \le 4|H|$, these two polynomials agree on at most $4|H|$ values. Let

$$p_C(x) \triangleq C(x, p_A(\textsc{Aff}(x)), \ldots, p_A(\textsc{Aff}_9(x))).$$

By a union bound, $p_C(x)$ is 10δ-close to the evaluation of $P_C(x)$ on $\mathbb{F}$. We conclude the fraction of inputs on which p_C and p_B agree, is at most $10\delta + \frac{4|H|}{|\mathbb{F}|} \le 1/2$. Thus, the third test of our verifier will reject with probability at least half. The proof of Theorem 1 is complete.

6. AN ALTERNATIVE TO THE "SUM-CHECK" PROTOCOL

6.1 Preliminaries - PCPPs for Reed-Muller Codes

It is possible to extend the PCPP for the RS-code into one for the Reed-Muller code (based on multivariate polynomials), given the extensive literature on testing multivariate polynomials using axis parallel lines [4, 5, 15, 3, 27, 16]. Let $\mathrm{RM}(\mathbb{F}, S, d, m)$ be the m-variate Reed-Muller code with degree bound d, evaluated at S^m, i.e. $\mathrm{RM}(\mathbb{F}, S, d, m) = \left\{ \langle Q(x_1, \ldots, x_m) \rangle_{x_1 \leftarrow S, \ldots, x_m \leftarrow S} : \forall i \in [m], \deg_{x_i}(Q) \le d \right\}$. For a set $S \subseteq \mathbb{F}$ and m-variate function $f : S^m \to \mathbb{F}$, let $\delta_m^d(f)$ be the fractional distance of f from $\mathrm{RM}(\mathbb{F}, S, d, m)$. Let $\delta_{m,i}^d(f)$ denote the fractional distance of f from a polynomial of degree d in the ith variable, and unbounded degree in all other variables. Finally, let $\mathbb{E}[\delta_{m,i}^d(f)]$ be the expectation of $\delta_{m,i}^d$ over random $i \in [m]$. The following Lemma is a rephrasing of [3, Lemma 5.2.1].

LEMMA 10. *[3] There exists a universal constant c such that for every $S \subset \mathbb{F}$ such that $|S| \ge \mathrm{poly}(m, d)$,*

$$\delta_m^d(f) \le c \cdot m \cdot \mathbb{E}[\delta_{m,i}^d(f)]$$

Lemma 10 together with Theorem 4 imply efficient PCPPs for Reed-Muller codes. A proof of the following Lemma will appear in the full version.

LEMMA 11 (RM PCPP). *Let $S \subset \mathbb{F}$ and d, m be integers such that $|S| \ge poly(m, d)$ for the polynomial of Lemma 10. If $\mathrm{RS}(\mathbb{F}, S, d)$ has a PCPP with length ℓ, query complexity q, randomness r and soundness $s(\delta)$, then the Reed-Muller Code $\mathrm{RM}(\mathbb{F}, S, d, m)$ has a PCPP with length $\le m \cdot n^{m-1} \cdot \ell$, query complexity q, randomness $\log(m \cdot n^{m-1}) + r$ and soundness $\ge s(\delta)/m$.*

REMARK 12. *A more query efficient test can be constructed when $S = \mathbb{F}$. Instead of axis parallel lines, we use an ϵ-biased set of directions as in [11]. This results in proofs of similar length and query complexity and slightly larger randomness, but the soundness is as large as $\Omega(s(\delta))$ (and independent of m).*

6.2 Multivariate Zero Testing

The multi-variate version of the zero-testing problem is crucial to all previous algebraic PCP constructions [5, 3, 2, 27, 20, 18, 11, 9], and was solved using the sum-check protocol of [25]. In the multivariate problem we are given sets $S, H \subset \mathbb{F}$ and oracle access to a multivariate function $f : S^m \to \mathbb{F}$. We are asked to verify f is close to a polynomial of degree $\le d$ in each variable that evaluates to zero on H^m (once again, we do not need to assume $H \subset S$). Let $\mathrm{RM}_H(\mathbb{F}, S, d, m)$ be the sub-code of the Reed-Muller code consisting of all (evaluations of) polynomials that vanish on H^m. We prove the following generalization of Lemma 7

LEMMA 13 (MULTIVARIATE ZERO TESTING). *Suppose that* $\mathrm{RM}(\mathbb{F}, S, d, m)$ *has a PCPP of length* ℓ*, query complexity* q*, randomness* r *and soundness* $s(\delta)$*. Then for any* $H \subset \mathbb{F}$*,* $\mathrm{RM}_H(\mathbb{F}, S, d, m)$ *has a PCPP with* proof length $m \cdot |S|^m + (m+1)\ell$*,* randomness $\max\{r, m \log |S|\}$*,* query complexity $(m+1)(q+1)$*, and* soundness $\geq \min\{s, 1 - ((m+1)\delta + \left(\frac{d}{|S|}\right)^m)\}$.

Notice the query complexity of previous solutions to this problem depended also on the size of H. Our solution has query complexity that depends only on m and is based on a straightforward characterization of RM_H (similar to Alon's Combinatorial Nullstellensatz [1]).

The catch in immediately extending the univariate verifier of Lemma 7 to even the bivariate case is that the "factoring" concept does not extend immediately. Specifically, if we are given that a bivariate polynomial $Q(x, y)$ has a zero at (α, β) this does not imply that $Q(x, y)$ has some nice factors. However, one can abstract a nice property about Q from this zero. Specifically, we can say that there exist polynomials $A(x, y), B(x, y)$ (of the right degree) such that $Q(x, y) = A(x, y) \cdot (x - \alpha) + B(x, y) \cdot (x - \beta)$. Thus to prove that $Q(\alpha, \beta) = 0$, we may ask the prover to give oracles for $Q(x, y)$. $A(x, y)$ and $B(x, y)$. We can then test that Q, A and B are of low-degree and that they satisfy the identity above. Extending this idea to m-variate polynomials that are zero on an entire generalized rectangle is straightforward. The technical lemma giving the identity is included below. (The lemma is also a key ingredient in Alon's "Combinatorial Nullstellensatz" [1].)

LEMMA 14. *Let* $Q(x_1, \ldots, x_m)$ *be a polynomial over* $\mathbb{F}_Q$ *of degree* d *in each of* m *variables. Let* $H \subseteq \mathbb{F}_Q$ *and let* $g_H(z) \stackrel{\text{def}}{=} \prod_{\beta \in H}(z - \beta)$*. Then* Q *evaluates to zero on* H^m *iff there exist* m*-variate polynomials* $A_1, \ldots, A_m$ *of individual degree at most* d *such that* $Q(\vec{x}) = \sum_{i=1}^m A_i(\vec{x}) \cdot g_H(x_i)$.

REMARK 15. *The lemma above is intentionally sloppy with degree bounds. While tighter degree bounds on* A_i*'s can be obtained, this won't be needed for our PCPs.*

PROOF. One direction is immediate. If $Q(\vec{x}) = \sum_{i=1}^m A_i(\vec{x}) \cdot g_H(x_i)$ then $Q(\vec{\alpha}) = 0$ for every $\vec{\alpha} \in H^m$. The other direction is proved in three steps. First, we show that for any polynomial $P(x_1, \ldots, x_m)$ of degree d_j in x_j, and any $i \in \{1, \ldots, m\}$, there exist polynomials $B(x_1, \ldots, x_m)$ and $C(x_1, \ldots, x_m)$ of degree at most d_j in x_j, with the degree of C in x_i being at most $\min\{d_j, |H| - 1\}$, such that $P(\vec{x}) = B(\vec{x}) \cdot g_H(x_i) + C(\vec{x})$. Second, we show that there exist polynomials $A_1, \ldots, A_m$ and R with the A_i's having degree at most d in each variable and R having degree at most $|H| - 1$ in each variable such that $Q(\vec{x}) = \sum_{i=1}^m A_i(\vec{x}) \cdot g_H(x_i) + R(\vec{x})$ (where Q is the polynomial from the lemma statement). In the final step, we show that $R(\vec{x}) = 0$, concluding the proof.

STEP 1: Recall that any polynomial $f(x_i)$ can be written as $q(x_i) \cdot g_H(x_i) + r(x_i)$ where r has degree less than $|H|$. Applying this fact to the monomials x_i^D (for non-negative D) we find that there exist polynomials $q_D(x_i)$ and $r_D(x_i)$, with degree of q_D being at most D and degree of r_D being less than $|H|$, such that $x_i^D = q_D(x_i) \cdot g_H(x_i) + r_D(x_i)$. Now consider any polynomial $P(x_1, \ldots, x_m)$ of degree d_i in x_i. Suppose $P(\vec{x}) = \sum_{D=0}^{d_i} P_i(\vec{x}') \cdot x^D$, where $\vec{x}' = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_m)$. Writing the monomials x_i^D in terms of the q_D's and r_D's, we get:

$$P(\vec{x}) = \left(\sum_{D=0}^{d_i} P_i(\vec{x}') q_D(x_i)\right) \cdot g_H(x_i) + \left(\sum_{D=0}^{d_i} P_i(\vec{x}') r_D(x_i)\right).$$

Letting $B(\vec{x}) = \sum_{D=0}^{d_i} P_i(\vec{x}') q_D(x_i)$ and $C(\vec{x}) = \left(\sum_{D=0}^{d_i} P_i(\vec{x}') r_D(x_i)\right)$ yields the polynomials as claimed. (In particular the degrees of B and C in any variable are no more than of P, and the degree of C in x_i is smaller than $|H|$.)

STEP 2: We now claim that there exist polynomials $A_1, \ldots, A_m$ and $R_0, \ldots, R_m$ such that for every $j \in \{0, \ldots, m\}$, $Q(\vec{x}) = \sum_{i=0}^j A_i(\vec{x}) \cdot g_H(x_i) + R_j(\vec{x})$, with A_i's being of degree at most d in each variable and R_j being of degree less than $|H|$ in $x_1, \ldots, x_j$ and of degree at most d in the remaining variables. The proof is straightforward by induction on j, with the induction step using Step 1 on the polynomial $P() = R_j()$ and the variable x_{j+1}. The final polynomials $A_1, \ldots, A_m$ and $R = R_m$ are the polynomials as required to yield the sub-claim of this step.

STEP 3: Finally we note that for every $\vec{\alpha} \in H^m$, we have $R(\vec{\alpha}) = Q(\vec{\alpha}) - \sum_{i=1}^m A_i(\vec{\alpha}) \cdot g_H(\alpha_i) = 0 - \sum_{i=1}^m 0 = 0$. But R is a polynomial of degree less than $|H|$ in each variable and is zero on the entire box H^m. This can only happen if $R \equiv 0$. Thus we get that $Q(\vec{x}) = \sum_{i=1}^m A_i(\vec{x}) \cdot g_H(x_i)$, with A_i's being of degree at most d in each variable, as required in the completeness condition. ∎

Lemma 14, combined with the multivariate polynomial verifier from Lemma 11 prove Lemma 13.

PROOF. (of Multivariate Zero Testing Lemma 13): As a proof of the proximity of $q \in \mathbb{F}^{S^m}$ to the code $\mathrm{RM}_H(\mathbb{F}, S, d, m)$ our verifier expects *(i)* the evaluations of $A_1, \ldots, A_m$ from Lemma 14 on S^m (denoted $a_1, \ldots, a_m$) and *(ii)* for each of $q, a_1, \ldots, a_m$, a proof of proximity of A_i to $\mathrm{RM}(\mathbb{F}, S, d, m)$. Proof length is as claimed. The verifier operates as follows. First, it tests proximity of each of $q, a_1, \ldots, a_m$ to $\mathrm{RM}(\mathbb{F}, S, d, m)$. Then, a random $\langle \alpha_1, \ldots, \alpha_m \rangle \in S^m$ is selected and verifier accepts iff $q(\vec{\alpha}) = \sum_{i=1}^m g_H(\alpha_i) \cdot a_i(\vec{\alpha})$. The query complexity is as claimed. Completeness follows from Lemma 14. As to the soundness, if any of $q, a_1, \ldots, a_m$ is δ-far from $\mathrm{RM}(\mathbb{F}, S, d, m)$ Verifier rejects with probability $s(\delta)$. Otherwise, q is δ-close to a polynomial Q that doesn't vanish on H^m. If $A_1, \ldots, A_m$ are the polynomials closest to $a_1, \ldots, a_m$ respectively, then by Lemma 14 we get $Q(\vec{x}) \neq \sum_i A_i(\vec{x}) \cdot g_H(x_i)$ and Q has degree at most $2d$ in each variable. Thus, the two polynomials agree on $\leq (2d)^m$ points so the acceptance probability of Verifier is $\leq (m+1)\delta + \left(\frac{d}{|S|}\right)^m$ as claimed. ∎

7. ACKNOWLEDGMENTS

We thank Oded Goldreich, Prahladh Harsha, Salil Vadhan and Chris Umans for helpful discussions. We thank Don Coppersmith for pointing us to Linnik's Theorem. We thank Venkat Guruswami and Subhash Khot for pointing out errors in a previous version of the paper.

8. REFERENCES

[1] ALON, N. Combinatorial Nullstellensatz. *Combinatorics, Probability and Computing*, 8 (1999), 7-29.

[2] Arora, S., Lund, C., Motwani, R., Sudan, M., and Szegedy, M. Proof verification and the hardness of approximation problems. *Journal of the ACM 45*, 3 (May 1998), 501–555. (Preliminary Version in *33rd FOCS*, 1992).

[3] Arora, S., and Safra, S. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM 45*, 1 (Jan. 1998), 70–122. (Preliminary Version in *33rd FOCS*, 1992).

[4] Babai, L., Fortnow, L. and Lund, C. Nondeterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1(1):3-40, 1991.

[5] Babai, L., Fortnow, L., Levin, L. A., and Szegedy, M. Checking computations in polylogarithmic time. In *Proc.* 23*rd ACM Symp. on Theory of Computing* (New Orleans, Louisiana, 6–8 May 1991), pp. 21–31.

[6] Barak, B. How to go beyond the black-box simulation barrier. In *Proc.* 42*nd IEEE Symp. on Foundations of Comp. Science* (Las Vegas, Nevada, 14–17 Oct. 2001), pp. 106–115.

[7] Bellare, M., Goldreich, O., and Sudan, M. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal of Computing 27*, 3 (June 1998), 804–915. (Preliminary Version in *36th FOCS*, 1995).

[8] Bellare, M., Goldwasser, S., Lund, C., and Russell, A. Efficient probabilistically checkable proofs and applications to approximation. In *Proc.* 25*th ACM Symp. on Theory of Computing* (San Diego, California, 16–18 May 1993), pp. 294–304.

[9] Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M. and Vadhan, S. Robust PCPs of Proximity, Shorter PCPs and Applications to Coding. In *STOC 2004*, Chicago.

[10] Ben-Sasson, E. and Sudan, M. Simple PCPs with Poly-log Rate and Query Complexity. Preliminary version at `http://eccc.uni-trier.de/eccc-reports/2004/TR04-060/index.html`

[11] Ben-Sasson, E., Sudan, M., Vadhan, S., and Wigderson, A. Randomness-efficient low degree tests and short PCPs via epsilon-biased sets. In *Proc.* 35*th ACM Symp. on Theory of Computing* (San Diego, California, 9–11 June 2003), pp. 612–621.

[12] Canetti, R., Goldreich, O., and Halevi, S. The random oracle methodology, revisited. In *Proc.* 30*th ACM Symp. on Theory of Computing* (Dallas, Texas, 23–26 May 1998), pp. 209–218.

[13] Dinur, I., Fischer, E., Kindler, G., Raz, R., and Safra, S. PCP Characterizations of NP: Towards a Polynomially-Small Error-Probability In *Proc.* 31*st ACM Symp. on Theory of Computing*, 1999, pp. 29-40

[14] Dinur, I., and Reingold, O. PCP testers: Towards a more combinatorial proof of PCP theorem. FOCS 2004.

[15] Feige, U., Goldwasser, S., Lovász, L., Safra, S., and Szegedy, M. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM 43*, 2 (Mar. 1996), 268–292. (Preliminary version in *32nd FOCS*, 1991).

[16] K. Friedl, Z. Hatsagi, and A. Shen. Low-degree tests. *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 57–64, 1994.

[17] Goldreich, O. A sample of samplers – a computational perspective on sampling. Tech. Rep. TR97-020, Electronic Colloquium on Computational Complexity, 1997.O. Goldreich

[18] Goldreich, O., and Sudan, M. Locally testable codes and PCPs of almost linear length. In *Proc.* 43*rd IEEE Symp. on Foundations of Comp. Science* (Vancouver, Canada, 16–19 Nov. 2002), pp. 13–22. (See ECC Report TR02-050, 2002).

[19] Guruswami, V., Lewin, D., Sudan, M., and Trevisan, L. A tight characterization of NP with 3-query PCPs. In *Proc.* 39*th IEEE Symp. on Foundations of Comp. Science* (Palo Alto, California, 8–11 Nov. 1998), pp. 18–27.

[20] Harsha, P., and Sudan, M. Small PCPs with low query complexity. *Computational Complexity 9*, 3–4 (Dec. 2000), 157–201. (Preliminary Version in *18th STACS*, 2001).

[21] Håstad, J. Some optimal inapproximability results. *Journal of the ACM 48*, 4 (July 2001), 798–859. (Preliminary Version in *29th STOC*, 1997).

[22] Kilian, J. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proc.* 24*th ACM Symp. on Theory of Computing* (Victoria, British Columbia, Canada, 4–6 May 1992), pp. 723–732.

[23] Lidl, R. and Niederreiter, H *Introduction to finite fields and their applications*. Cambridge University Press, Cambridge, UK. Revised edition, 1994.

[24] Linnik, U. V. On the Least Prime in an Arithmetic Progression. I. The Basic Theorem. In *Mat. Sbornik N. S.* 15 (57), 139-178, 1944.

[25] Lund, C., Fortnow, L., Karloff, H., and Nisan, N. Algebraic methods for interactive proof systems. In *Proc.* 31*st IEEE Symp. on Foundations of Comp. Science* (St. Louis, Missouri, 22–24 Oct. 1990), pp. 2–10.

[26] Micali, S. Computationally sound proofs. *SIAM Journal of Computing 30*, 4 (2000), 1253–1298. (Preliminary Version in *35th FOCS*, 1994).

[27] Polishchuk, A., and Spielman, D. A. Nearly-linear size holographic proofs. In *Proc.* 26*th ACM Symp. on Theory of Computing* (Montréal, Québec, Canada, 23–25 May 1994), pp. 194–203.

[28] Rubinfeld, R., and Sudan, M. Robust characterizations of polynomials with applications to program testing. *SIAM Journal of Computing 25*, 2 (Apr. 1996), 252–271. (Preliminary Version in *23rd STOC*, 1991 and *3rd SODA*, 1992).

[29] Samorodnitsky, A., and Trevisan, L. A PCP characterization of NP with optimal amortized query complexity. In *Proc.* 32*nd ACM Symp. on Theory of Computing* (Portland, Oregon, 21–23 May 2000), pp. 191–199.

[30] Spielman, D. A. Computationally Efficient Error-Correcting Codes and Holographic Proofs Ph. D. Thesis, MIT, Cambridge, MA, 1995.